**http接口自动化测试框架实现**

作者：张元礼

<http://blog.csdn.net/vincetest>

一、测试需求描述

对服务后台一系列的http接口功能测试。

输入：根据接口描述构造不同的参数输入值

输出：XML文件

eg:http://xxx.com/xxx\_product/test/content\_book\_list.jsp?listid=1

二、实现方法

1、选用Python脚本来驱动测试

2、采用Excel表格管理测试数据，包括用例的管理、测试数据录入、测试结果显示等等，这个需要封装一个Excel的类即可。

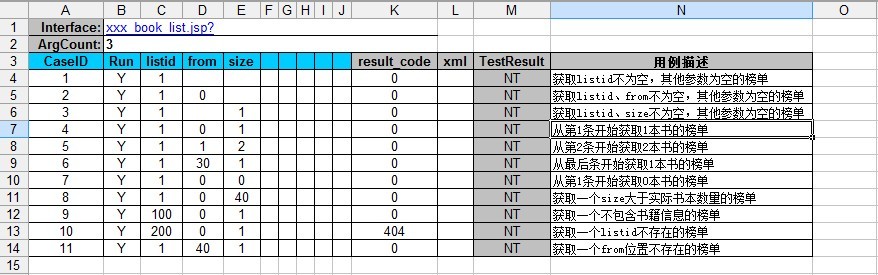
3、调用http接口采用Python封装好的API即可

4、测试需要的http组装字符转处理即可

5、设置2个检查点，XML文件中的返回值字段（通过解析XML得到）；XML文件的正确性（文件对比）

6、首次执行测试采用半自动化的方式，即人工检查输出的XML文件是否正确，一旦正确将封存XML文件，为后续回归测试的预期结果，如果发现错误手工修正为预期文件。（注意不是每次测试都人工检查该文件，只首次测试的时候才检查）

三、Excel表格样式



四、实现代码（代码才是王道，有注释很容易就能看明白的）

1、测试框架代码

**[python]** [view plaincopy](http://blog.csdn.net/vincetest/article/details/6341658)

1. #\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
2. # TestFrame.py
3. # Author     : Vince
4. # Version    : 1.1.2
5. # Date       : 2011-3-14
6. # Description: 自动化测试平台
7. #\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
9. **import** os,sys, urllib, httplib, profile, datetime, time
10. **from** xml2dict **import** XML2Dict
11. **import** win32com.client
12. **from** win32com.client **import** Dispatch
13. **import** xml.etree.ElementTree as et
14. #import MySQLdb
16. #Excel表格中测试结果底色
17. OK\_COLOR=0xffffff
18. NG\_COLOR=0xff
19. #NT\_COLOR=0xffff
20. NT\_COLOR=0xC0C0C0
22. #Excel表格中测试结果汇总显示位置
23. TESTTIME=[1, 14]
24. TESTRESULT=[2, 14]
26. #Excel模版设置
27. #self.titleindex=3        #Excel中测试用例标题行索引
28. #self.casebegin =4        #Excel中测试用例开始行索引
29. #self.argbegin   =3       #Excel中参数开始列索引
30. #self.argcount  =8        #Excel中支持的参数个数
31. **class** create\_excel:
32. **def** \_\_init\_\_(self, sFile, dtitleindex=3, dcasebegin=4, dargbegin=3, dargcount=8):
33. self.xlApp = win32com.client.Dispatch('et.Application')   #MS:Excel  WPS:et
34. **try**:
35. self.book = self.xlApp.Workbooks.Open(sFile)
36. **except**:
37. print\_error\_info()
38. **print** "打开文件失败"
39. exit()
40. self.file=sFile
41. self.titleindex=dtitleindex
42. self.casebegin=dcasebegin
43. self.argbegin=dargbegin
44. self.argcount=dargcount
45. self.allresult=[]
47. self.retCol=self.argbegin+self.argcount
48. self.xmlCol=self.retCol+1
49. self.resultCol=self.xmlCol+1
51. **def** close(self):
52. #self.book.Close(SaveChanges=0)
53. self.book.Save()
54. self.book.Close()
55. #self.xlApp.Quit()
56. **del** self.xlApp
58. **def** read\_data(self, iSheet, iRow, iCol):
59. **try**:
60. sht = self.book.Worksheets(iSheet)
61. sValue=str(sht.Cells(iRow, iCol).Value)
62. **except**:
63. self.close()
64. **print**('读取数据失败')
65. exit()
66. #去除'.0'
67. **if** sValue[-2:]=='.0':
68. sValue = sValue[0:-2]
69. **return** sValue
71. **def** write\_data(self, iSheet, iRow, iCol, sData, color=OK\_COLOR):
72. **try**:
73. sht = self.book.Worksheets(iSheet)
74. sht.Cells(iRow, iCol).Value = sData.decode("utf-8")
75. sht.Cells(iRow, iCol).Interior.Color=color
76. self.book.Save()
77. **except**:
78. self.close()
79. **print**('写入数据失败')
80. exit()
82. #获取用例个数
83. **def** get\_ncase(self, iSheet):
84. **try**:
85. **return** self.get\_nrows(iSheet)-self.casebegin+1
86. **except**:
87. self.close()
88. **print**('获取Case个数失败')
89. exit()
91. **def** get\_nrows(self, iSheet):
92. **try**:
93. sht = self.book.Worksheets(iSheet)
94. **return** sht.UsedRange.Rows.Count
95. **except**:
96. self.close()
97. **print**('获取nrows失败')
98. exit()
100. **def** get\_ncols(self, iSheet):
101. **try**:
102. sht = self.book.Worksheets(iSheet)
103. **return** sht.UsedRange.Columns.Count
104. **except**:
105. self.close()
106. **print**('获取ncols失败')
107. exit()
109. **def** del\_testrecord(self, suiteid):
110. **try**:
111. #为提升性能特别从For循环提取出来
112. nrows=self.get\_nrows(suiteid)+1
113. ncols=self.get\_ncols(suiteid)+1
114. begincol=self.argbegin+self.argcount
116. #提升性能
117. sht = self.book.Worksheets(suiteid)
119. **for** row **in** range(self.casebegin, nrows):
120. **for** col **in** range(begincol, ncols):
121. str=self.read\_data(suiteid, row, col)
122. #清除实际结果[]
123. startpos = str.find('[')
124. **if** startpos>0:
125. str = str[0:startpos].strip()
126. self.write\_data(suiteid, row, col, str, OK\_COLOR)
127. **else**:
128. #提升性能
129. sht.Cells(row, col).Interior.Color = OK\_COLOR
130. #清除TestResul列中的测试结果，设置为NT
131. self.write\_data(suiteid, row,  self.argbegin+self.argcount+1, ' ', OK\_COLOR)
132. self.write\_data(suiteid, row, self.resultCol, 'NT', NT\_COLOR)
133. **except**:
134. self.close()
135. **print**('清除数据失败')
136. exit()
138. #执行调用
139. **def** HTTPInvoke(IPPort, url):
140. conn = httplib.HTTPConnection(IPPort)
141. conn.request("GET", url)
142. rsps = conn.getresponse()
143. data = rsps.read()
144. conn.close()
145. **return** data
147. #获取用例基本信息[Interface,argcount,[ArgNameList]]
148. **def** get\_caseinfo(Data, SuiteID):
149. caseinfolist=[]
150. sInterface=Data.read\_data(SuiteID, 1, 2)
151. argcount=int(Data.read\_data(SuiteID, 2, 2))
153. #获取参数名存入ArgNameList
154. ArgNameList=[]
155. **for** i **in** range(0, argcount):
156. ArgNameList.append(Data.read\_data(SuiteID, Data.titleindex, Data.argbegin+i))
158. caseinfolist.append(sInterface)
159. caseinfolist.append(argcount)
160. caseinfolist.append(ArgNameList)
161. **return** caseinfolist
163. #获取输入
164. **def** get\_input(Data, SuiteID, CaseID, caseinfolist):
165. sArge=''
166. #参数组合
167. **for** j **in** range(0, caseinfolist[1]):
168. **if** Data.read\_data(SuiteID, Data.casebegin+CaseID, Data.argbegin+j) != "None":
169. sArge=sArge+caseinfolist[2][j]+'='+Data.read\_data(SuiteID, Data.casebegin+CaseID, Data.argbegin+j)+'&'
171. #去掉结尾的&字符
172. **if** sArge[-1:]=='&':
173. sArge = sArge[0:-1]
174. sInput=caseinfolist[0]+sArge    #组合全部参数
175. **return** sInput
177. #结果判断
178. **def** assert\_result(sReal, sExpect):
179. sReal=str(sReal)
180. sExpect=str(sExpect)
181. **if** sReal==sExpect:
182. **return** 'OK'
183. **else**:
184. **return** 'NG'
186. #将测试结果写入文件
187. **def** write\_result(Data, SuiteId, CaseId, resultcol, \*result):
188. **if** len(result)>1:
189. ret='OK'
190. **for** i **in** range(0, len(result)):
191. **if** result[i]=='NG':
192. ret='NG'
193. **break**
194. **if** ret=='NG':
195. Data.write\_data(SuiteId, Data.casebegin+CaseId, resultcol,ret, NG\_COLOR)
196. **else**:
197. Data.write\_data(SuiteId, Data.casebegin+CaseId, resultcol,ret, OK\_COLOR)
198. Data.allresult.append(ret)
199. **else**:
200. **if** result[0]=='NG':
201. Data.write\_data(SuiteId, Data.casebegin+CaseId, resultcol,result[0], NG\_COLOR)
202. **elif** result[0]=='OK':
203. Data.write\_data(SuiteId, Data.casebegin+CaseId, resultcol,result[0], OK\_COLOR)
204. **else**:  #NT
205. Data.write\_data(SuiteId, Data.casebegin+CaseId, resultcol,result[0], NT\_COLOR)
206. Data.allresult.append(result[0])
208. #将当前结果立即打印
209. **print** 'case'+str(CaseId+1)+':', Data.allresult[-1]
211. #打印测试结果
212. **def** statisticresult(excelobj):
213. allresultlist=excelobj.allresult
214. count=[0, 0, 0]
215. **for** i **in** range(0, len(allresultlist)):
216. #print 'case'+str(i+1)+':', allresultlist[i]
217. count=countflag(allresultlist[i],count[0], count[1], count[2])
218. **print** 'Statistic result as follow:'
219. **print** 'OK:', count[0]
220. **print** 'NG:', count[1]
221. **print** 'NT:', count[2]
223. #解析XmlString返回Dict
224. **def** get\_xmlstring\_dict(xml\_string):
225. xml = XML2Dict()
226. **return** xml.fromstring(xml\_string)
228. #解析XmlFile返回Dict
229. **def** get\_xmlfile\_dict(xml\_file):
230. xml = XML2Dict()
231. **return** xml.parse(xml\_file)
233. #去除历史数据expect[real]
234. **def** delcomment(excelobj, suiteid, iRow, iCol, str):
235. startpos = str.find('[')
236. **if** startpos>0:
237. str = str[0:startpos].strip()
238. excelobj.write\_data(suiteid, iRow, iCol, str, OK\_COLOR)
239. **return** str
241. #检查每个item （非结构体）
242. **def** check\_item(excelobj, suiteid, caseid,real\_dict, checklist, begincol):
243. ret='OK'
244. **for** checkid **in** range(0, len(checklist)):
245. real=real\_dict[checklist[checkid]]['value']
246. expect=excelobj.read\_data(suiteid, excelobj.casebegin+caseid, begincol+checkid)
248. #如果检查不一致测将实际结果写入expect字段，格式：expect[real]
249. #将return NG
250. result=assert\_result(real, expect)
251. **if** result=='NG':
252. writestr=expect+'['+real+']'
253. excelobj.write\_data(suiteid, excelobj.casebegin+caseid, begincol+checkid, writestr, NG\_COLOR)
254. ret='NG'
255. **return** ret
257. #检查结构体类型
258. **def** check\_struct\_item(excelobj, suiteid, caseid,real\_struct\_dict, structlist, structbegin, structcount):
259. ret='OK'
260. **if** structcount>1:  #传入的是List
261. **for** structid **in** range(0, structcount):
262. structdict=real\_struct\_dict[structid]
263. temp=check\_item(excelobj, suiteid, caseid,structdict, structlist, structbegin+structid\*len(structlist))
264. **if** temp=='NG':
265. ret='NG'
267. **else**: #传入的是Dict
268. temp=check\_item(excelobj, suiteid, caseid,real\_struct\_dict, structlist, structbegin)
269. **if** temp=='NG':
270. ret='NG'
272. **return** ret
274. #获取异常函数及行号
275. **def** print\_error\_info():
276. """Return the frame object for the caller's stack frame."""
277. **try**:
278. **raise** Exception
279. **except**:
280. f = sys.exc\_info()[2].tb\_frame.f\_back
281. **print** (f.f\_code.co\_name, f.f\_lineno)
283. #测试结果计数器，类似Switch语句实现
284. **def** countflag(flag,ok, ng, nt):
285. calculation  = {'OK':**lambda**:[ok+1, ng, nt],
286. 'NG':**lambda**:[ok, ng+1, nt],
287. 'NT':**lambda**:[ok, ng, nt+1]}
288. **return** calculation[flag]()

2、项目测试代码

**[python]** [view plaincopy](http://blog.csdn.net/vincetest/article/details/6341658)

1. # -\*- coding: utf-8 -\*-
2. #\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
3. # xxx\_server\_case.py
4. # Author     : Vince
5. # Version    : 1.0
6. # Date       : 2011-3-10
7. # Description: 内容服务系统测试代码
8. #\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
10. **from** testframe **import** \*
11. **from** common\_lib **import** \*
13. httpString='http://xxx.com/xxx\_product/test/'
14. expectXmldir=os.getcwd()+'/TestDir/expect/'
15. realXmldir=os.getcwd()+'/TestDir/real/'
17. **def** run(interface\_name, suiteid):
18. **print** '【'+interface\_name+'】' + ' Test Begin,please waiting...'
19. **global** expectXmldir, realXmldir
21. #根据接口名分别创建预期结果目录和实际结果目录
22. expectDir=expectXmldir+interface\_name
23. realDir=realXmldir+interface\_name
24. **if** os.path.exists(expectDir) == 0:
25. os.makedirs(expectDir)
26. **if** os.path.exists(realDir) == 0:
27. os.makedirs(realDir)
29. excelobj.del\_testrecord(suiteid)  #清除历史测试数据
30. casecount=excelobj.get\_ncase(suiteid) #获取case个数
31. caseinfolist=get\_caseinfo(excelobj, suiteid) #获取Case基本信息
33. #遍历执行case
34. **for** caseid **in** range(0, casecount):
35. #检查是否执行该Case
36. **if** excelobj.read\_data(suiteid,excelobj.casebegin+caseid, 2)=='N':
37. write\_result(excelobj, suiteid, caseid, excelobj.resultCol, 'NT')
38. **continue** #当前Case结束，继续执行下一个Case
40. #获取测试数据
41. sInput=httpString+get\_input(excelobj, suiteid, caseid, caseinfolist)
42. XmlString=HTTPInvoke(com\_ipport, sInput)     #执行调用
44. #获取返回码并比较
45. result\_code=et.fromstring(XmlString).find("result\_code").text
46. ret1=check\_result(excelobj, suiteid, caseid,result\_code, excelobj.retCol)
48. #保存预期结果文件
49. expectPath=expectDir+'/'+str(caseid+1)+'.xml'
50. #saveXmlfile(expectPath, XmlString)
52. #保存实际结果文件
53. realPath=realDir+'/'+str(caseid+1)+'.xml'
54. saveXmlfile(realPath, XmlString)
56. #比较预期结果和实际结果
57. ret2= check\_xmlfile(excelobj, suiteid, caseid,expectPath, realPath)
59. #写测试结果
60. write\_result(excelobj, suiteid, caseid, excelobj.resultCol, ret1, ret2)
61. **print** '【'+interface\_name+'】' + ' Test End!'

3、测试入口

**[python]** [view plaincopy](http://blog.csdn.net/vincetest/article/details/6341658)

1. # -\*- coding: utf-8 -\*-
2. #\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
3. # main.py
4. # Author     : Vince
5. # Version    : 1.0
6. # Date       : 2011-3-16
7. # Description: 测试组装，用例执行入口
8. #\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
10. **from** testframe **import** \*
11. **from** xxx\_server\_case **import** \*
12. **import** xxx\_server\_case
14. #产品系统接口测试
15. #设置测试环境
16. xxx\_server\_case.excelobj=create\_excel(os.getcwd()+'/TestDir/xxx\_Testcase.xls')
17. xxx\_server\_case.com\_ipport=xxx.com'
19. #Add testsuite begin
20. run("xxx\_book\_list", 4)
21. #Add other suite from here
22. #Add testsuite end
24. statisticresult(xxx\_server\_case.excelobj)
25. xxx\_server\_case.excelobj.close()

最后感谢我的同事Roger为此做了一些优化，后续优化的东东还很多，我们一直在努力！

欢迎转载此文，转载时请注明文章来源：张元礼的博客 <http://blog.csdn.net/vincetest>